METHOD AND APPARATUS FOR PREVENTING UN-AUTHORIZED ATTACHMENT OF COMPUTER PERIPHERALS

Field of the Invention

[0001] The present invention relates to computers in general, and, more particularly, to preventing un-authorized attachment of computer peripherals to a computer or computer network.

Background of the Invention

[0002] FIG. 1 depicts block diagram of a portion of a typical computer network in the prior art. Computer network 102 comprises computer network bus 104 to which computer peripherals can be attached or shared across the network. In particular, host computer 106 via computer port 108 is attached to computer network bus 104 via computer network port 110. Peripheral 118 is directly connected to the computer network via peripheral port 120 and computer network port 122. Alternatively, peripheral 114 is directly connected to host computer 106 via peripheral port 116 and host computer port 112.

[0003] If such a network is a classified network, provision must be made to track configuration changes, particularly the un-authorized attachment of computer peripherals. If this is not done, then un-authorized data and "viruses" residing in software/firmware contained within computer peripherals attached to a host computer or computer network can reap havoc upon the computer or network or both. The expense and time required to track configuration changes manually is high. Manual tracking often leads to long delays and waits when changes must be made in a classified development environment.

[0004] Automated configuration control and logging of attachments of computer peripherals is superior to manual tracking. The prior art, however, concentrates primarily on enhanced automated software protection of computer assets. In U.S. Pat. Nos. 5,144,659 and 5,289,540, Jones discloses a security method wherein a hard drive controller provides extra security functions. In U.S. Pat. No. 5,434,562, Reardon discloses the use of CPU-independent, user activated key lock switches by which a CPU-independent security controller can be configured and reprogrammed in a secure fashion.

Summary f the Invention

[0005] The illustrative embodiments of the present invention address the problem of un-authorized attachment of computer peripherals to a computer network/host computer. Common to the illustrative embodiments are two, mating, keyed-adapters that couple the host computer or computer network to the peripheral. In some embodiments, one of the keyed-adapters contains hardware that a host computer accesses to retrieve a unique identifier from the adapter associated with the attached peripheral. Software residing in the host computer will attempt to match the unique identifier to a list of authorized identifiers. If no match is found, then either software residing within the host computer or hardware within one or the other adapter under software control prevents communication to or from the attached peripheral.

[0006] In the first illustrative embodiment, the adapters are passive.

[0007] In the second illustrative embodiment, the peripheral-side adapter contains hardware for storing a unique identifier corresponding to the attached peripheral. The host computer/processor retrieves this stored identifier and compares it to a list of authorized identifiers. If a match is found, then communication is enabled in software so that software processes can communicate with the peripheral. If a match is not found, then, by default, communication to and from the peripheral is not allowed. Software also resides in the host computer/processor to log mismatches and to notify security personnel or a system administrator of the attempt to attach the unauthorized peripheral.

[0008] The third illustrative embodiment contains all the functionality of the second illustrative embodiment, except that control of communications to and from the peripheral is via a second hardware contained within the peripheral-side adapter under software control.

[0009] In the fourth illustrative embodiment, communication is initially enabled by a second hardware contained within the host computer/processor/network-side adapter under software control. If a match of an identifier stored in a first hardware within the peripheral-side adapter is not found in the database stored in memory on the host computer, then the second hardware is directed to disable communications to and from the peripheral.

[0010] The illustrative embodiment comprises: a first adapter, wherein the first adapter contains hardware for storing a unique identifier; a second adapter, wherein the first adapter couples a first port associated with a computer peripheral to the second adapter, and wherein the second adapter couples the first adapter to a second port associated with a processor; a first software module associated with the processor, wherein the first software module consults a list of identifiers within the first software module and wherein each of the identifiers is associated with a respective computer peripheral authorized for use with the processor; and, means for enabling communication to and from the computer peripheral under control of the processor.

Brief Description of the Drawings

[0011] FIG. 1 depicts block diagram of a portion of a typical computer network in the prior art.

[0012] FIG. 2 depicts a block diagram of the use of the illustrative embodiments of the present invention for the case of attaching a peripheral device directly to a host computer port.

[0013] FIG. 3 depicts a block diagram of the use of illustrative embodiments of the present invention for the case of attaching a peripheral device to a computer network port.

[0014] FIG. 4 depicts a block diagram of the first illustrative embodiment of the present invention.

[0015] FIG. 5 depicts keyed-connectors 404 and 408 comprising a tamper-proof seal.

[0016] FIG. 6 depicts keyed-connectors 404 and 408 comprising a screw head.

[0017] FIG. 7 depicts keyed-connectors 404 and 408 comprising a physical key.

[0018] FIG. 8 depicts a block diagram of host computer 106.

[0019] FIG. 9 depicts a block diagram of the second illustrative embodiment of the present invention.

[0020] FIG. 10 depicts the contents of memory 804 for the second illustrative embodiment of the present invention.

[0021] FIG. 11 is a flow chart of the actions performed by processor 802 as represented by the contents of second software module 1004.

[0022] FIG. 12 depicts the contents of first software module 1002.

[0023] FIG. 13 depicts the contents of identifier 1206.

[0024] FIG. 14 depicts a block diagram of the third illustrative embodiment of the present invention.

[0025] FIG. 15 depicts the contents of memory 804 for the third illustrative embodiment of the present invention.

[0026] FIG. 16 is a flow chart of the actions taken by processor 802 as represented by the contents of second software module 1504.

[0027] FIG. 17 depicts a block diagram of the fourth illustrative embodiment of the present invention.

[0028] FIG. 18 depicts the contents of memory 804 for the fourth illustrative embodiment of the present invention.

[0029] FIG. 19 is a flow chart of the actions taken by processor 802 as represented by the contents of second software module 1804.

Detailed Description

[0030] FIG. 2 depicts a block diagram of the use of the illustrative embodiments of the present invention for the case of attaching a peripheral device directly to a host computer port. Host computer 106 is attached to adapter 204 via host computer port 112. Adapter 202 is attached to peripheral device 114 via peripheral port 116. Adapter 202 mates with adapter 204. At least one of adapters 202 and 204 is permanently attached to its host (i.e., 202 to peripheral 114 or 204 to host computer 106).

[0031] FIG. 3 depicts a block diagram of the use of illustrative embodiments of the present invention for the case of attaching a peripheral device to a computer network port. Computer network 102 is connected via network bus 104 to various computer assets. The network is associated with host computer 106 via host computer port 108 and network port 110. Peripheral 118 is attached to adapter 302 via peripheral port 120. Adapter 304 of mates with adapter 302 and is attached to computer network bus 104 via network port 122. At least one of adapters 302 and is permanently attached to its host (i.e., 302 to peripheral 114 or 304 to host computer 106).

[0032] FIG. 4 depicts a block diagram of the first illustrative embodiment of the present invention. Adapter 402 comprises connector 404 which is keyed to connector 408 of adapter 406. In the first embodiment, the adapters are passive; in some other embodiments, one or both adapters 402 and 406 contain electronic and/or mechanical hardware. The amount of unique identifiers required depends on the number of peripherals attached (or could be attached) to host computer 106 or computer network 102 and also depends on the size of the network or the needs of the company using the equipment or both. In some cases, each peripheral will have a unique identifier. In other cases, one unique identifier per peripheral type (per printer type, modem type, *etc.*), or single identifier for one computer or network or both is all that is required. In still other cases, all that is required is one identifier for all products manufactured by the company.

[0033] FIG. 5 depicts keyed-connectors 404 and 408 comprising a tamper-proof seal. As shown in FIG. 5a, keyed-connector 404 comprises shaft 502, head 504, and electrical contacts 506-1 and 506-2, forming a male keyed-connector. Keyed-connector 408 comprises constriction members 508-1 and 508-2, receiving region 510, and electrical contacts 512-1 and 512-2, forming a female keyed-connector.

[0034] FIGS. 5b and 5c illustrate the use of keyed-connectors 404 and 408. As depicted in FIG. 5b, keyed-connector 404 slides between constriction members 508-1 and 508-2, compressing head 504 to fit in the intervening space. In FIG. 5c, head 504 has entered receiving region 510, expanding back to its original shape. Electrical contacts 506-1 contacts electrical contact 512-1, while electrical contact 506-2 contacts electrical contact 512-2. If an attempt is made to remove keyed-connector 404 from keyed-connector 408, the shape of head 504 initially prevents removal. If significant force is applied, however, head 504 will break, destroying keyed-connector 404 and revealing evidence of tampering.

[0035] FIG. 6 depicts keyed-connectors 404 and 408 comprising a screw head. Keyed-connector 404 is a male connector comprising threaded-head 602. Threaded-head 602 screws into grooves 604 of female keyed-connector 408.

[0036] FIG. 7 depicts keyed-connectors 404 and 408 comprising a physical key. As shown in FIG. 7a, male keyed-connector 404 comprises shaft 702, locking mechanism 704, blocking element 706, and key 708. Female keyed-connector 408 comprises receiving region 710. Initially, blocking element 706 surrounds shaft 702 and is locked to shaft 702 via locking mechanism 704. In this initial state, keyed-connector 404 cannot be inserted into keyed-connector 408. When key 708 is inserted and turned in blocking element 706 and locking mechanism 704, blocking element 706 can be removed from shaft 702. As shown in FIG. 7b, shaft 702 can subsequently be inserted into receiving region 710.

[0037] FIG. 8 depicts a block diagram of host computer 106. Host computer 106 comprises processor 802, memory 804, host computer bus 806, bus interface 808, host computer port 108, and host computer port 112, interconnected as shown. Processor 802 is a general-purpose or special-purpose processor that is capable of performing the functionality described below and with respect to FIGS. 10 through 13, 15, 16, 18, and 19. In particular, processor 802 is capable of storing data into memory 804, retrieving data from memory 804, executing programs stored in memory 804, and receiving, transmitting, and controlling bus interface 808 using host computer bus 806. Incoming and outgoing messages are communicated via ports 108 and 112, bus interface 808, and host computer bus 806. It will be clear to those skilled in the art how to make and use processor 802, memory 804, host computer bus 806, bus interface 808, host computer port 108, and host computer port 112.

[0038] FIG. 9 depicts a block diagram of the second illustrative embodiment of the present invention. The peripheral-side adapter 902 contains hardware 910 for storing, via peripheral ports 116 or 120, a unique identifier for the peripheral with which the adapter is associated. In some embodiments, hardware 910 comprises a memory and supporting hardware. The unique identifier can comprise, without limitation, a serial number, or a serial number and peripheral type. It will be appreciated by those skilled in the art that the unique identifier is not limited to a serial number and peripheral type.

[0039] Paths 912 and 914 provide the electrical connections between:

(i) peripheral port 116 and host computer port 112 or
(ii) peripheral port 120 and computer network port 122

via connectors 904 and 908, respectively. Path 912 and 914 permit processor 802 to access the contents of hardware 910.

[0040] FIG. 10 depicts the contents of memory 804 for the second illustrative embodiment of the present invention. In the second illustrative embodiment, memory 804 contains:

(1) first software module 1002 comprising a list of identifiers associated with the peripherals that are authorized to connect to host computer 106 or computer network 102;

(2) second software module 1004 for retrieving the contents of first hardware 910 and performing one or more actions based upon said contents; and,

(3) third software module 1006 for enabling communication to/from peripheral 114 or 118.

[0041] FIG. 11 is a flow chart of the actions performed by processor 802 as represented by the contents of second software module 1004. At task 1102, processor 802 retrieves the contents of hardware 910 either upon:

(i) boot up of host computer 106;
(ii) attachment of peripheral 118 to network port 122; or,
(iii) attachment of peripheral 114 to host computer port 112.

[0042] At task 1104, processor 802 compares the unique identifier with a list of identifiers contained within first software module 1002.

[0043] At task 1106, processor 802 determines whether a match is found in the list.

[0044] If a match is found, at task 1108, processor 802 enables communication between software modules or processes residing either in memory 804 or other devices connected to network 102 and the peripheral in question (*i.e.*, peripheral 114 or 118) via third software module 1006. Third software module 1006 by default disables communication with peripherals. If a match is not found, then optional tasks 1110 and 1112 are executed.

[0045] At task 1110, processor 802 stores information indicative of a failure to find a match of the retrieved unique identifier to elements in the list of software module 1002. This information can take the form of logging the attempt in memory 804, what identifier was retrieved, and supporting data for an investigation.

[0046] At task 1112, processor 802 generates an email indicative of a failure to find a match of the retrieved unique identifier retrieved to elements in the list of software module 1002. The email can be sent to the computer terminal of security officer if computer network 102 is a classified local area network, or to a system administrator.

[0047] FIG. 12 depicts the contents of first software module 1002. First software module 1002 comprises a list of identifiers for peripherals that are authorized for use with processor 802 such as identifiers 1202, 1204, and 1206.

[0048] FIG. 13 depicts the contents of identifier 1206. Identifier 1206 can be indicative of peripheral type 1302 (*e.g.,* a printer, I/O card, modem, *etc.*), along with a list of serial numbers 1304, 1306, and 1308 of peripherals that are allowed for use with processor 802 for given peripheral type 1302. It will be appreciated by those skilled in the art that identifier 1206 is not limited to the representation depicted in FIG. 13. How many elements identifier 1206 needs depends on the number of peripherals attached or could be attached to host computer 106 or computer network 102, the size of the network, or the needs of the company using the equipment. Some companies may require that each peripheral have a serial number, a serial number and peripheral type, a peripheral type, a single identifier for one computer or network or both, or one identifier for all products manufactured by the company.

[0049] FIG. 14 depicts a block diagram of the third illustrative embodiment of the present invention. Peripheral-side adapter 1402 contains first hardware 1410 for storing, via peripheral ports 116 or 120, a unique identifier of the peripheral with which the adapter is associated. First hardware 1410 comprises the same hardware as first hardware of FIG. 9. The unique identifier is the same as stored in first hardware 910 of FIG. 9.

[0050] Paths 1414 and 1416 provide the electrical connections between:

(i) peripheral port 116 and host computer port 112 or
(ii) peripheral port 120 and computer network port 122

via connectors 1404 and 1408. Path 1414 and 1416 permit processor 802 to access the contents of first hardware 1410. Inserted, however, in path 1414 of peripheral-side adapter 1402 is second hardware 1412. Second hardware 1412 provides, in hardware, the functionality described for third software module 1006. It comprises hardware that, by default, disables electrical communication between processor 802 and one of peripherals 114 and 118.

[0051] Second hardware 1412 is under the control of processor 802. Processor 802 retrieves the contents of first hardware 1410 either upon:

(i) boot up of host computer 106;

(ii) attachment of peripheral 118 to network port 122; or,
(iii) attachment of peripheral 114 to host computer port 112.

Processor 802 then compares the unique identifier with a list of identifiers of authorized peripherals. If a match is found, processor 802 sends commands to second hardware 1412 to permit electrical communication between one of processor 802 and network 102 and one of peripheral 114 and 118. Second hardware 1412 can comprise, for example, one or more normally-open relays, semiconductor switches, etc., and line-interrupt control. It will be appreciated by those skilled in the art how to make and use second hardware 1412.

[0052] FIG. 15 depicts the contents of memory 804 for the third illustrative embodiment of the present invention. In the third illustrative embodiment, memory 804 comprises first software module 1502 comprising a list of identifiers associated with peripherals that are authorized to be connected to host computer 106 or computer network 102; and second software module 1504 for retrieving the contents of first hardware 1410 and taking actions based upon said contents. In particular, the functionality provided by third software module 1006 of the first illustrative embodiment is replaced by the functionality of second hardware 1412. First software module 1502 has the same contents as first software module 1002 of the second illustrative embodiment.

[0053] FIG. 16 is a flowchart of the actions taken by processor 802 as represented by the contents of second software module 1504. At task 1602, processor 802 retrieves the contents of first hardware 1410 either upon:

(i) boot up of host computer 106;
(ii) attachment of peripheral 118 to network port 122; or,
(iii) attachment of peripheral 114 to host computer port 112.

[0054] At task 1604, processor 802 compares the unique identifier with a list of identifiers contained within a database of first software module 1502 stored in memory 804.

[0055] At task 1606, processor 802 decides whether a match is found in the list.

[0056] If a match is found, at task 1608, processor 802 enables communication between software modules or processes residing either in memory 804 or other devices connected to network 102 and the peripheral in question (*i.e.* either peripheral 114 or 118) via second hardware 1412. Second hardware 1412 by default disables communication with peripherals. Processor 802 sends commands directly to second hardware 1412 to enable communication. If a match is not found, then optional tasks 1610 and 1612 are executed.

[0057] At task 1610, processor 802 stores information indicative of a failure to find a match of the retrieved unique identifier to elements in the list of software module 1502. This information can take the form of logging the attempt in memory 804, what identifier was retrieved, and supporting data for an investigation.

[0058] At task 1612, processor 802 generates an email indicative of a failure to find a match of the retrieved unique identifier retrieved to elements in the list of software module 1602. The email can be sent to the computer terminal of security officer if computer network 102 is a classified local area network, or to a system administrator.

[0059] FIG. 17 depicts a block diagram of the fourth illustrative embodiment of the present invention. Peripheral-side adapter 1702 contains first hardware 1710 for storing, via peripheral ports 116 or 120, a unique identifier for the peripheral with which adapter is associated. First hardware 1710 comprises the same hardware as first hardware of FIG. 9. The unique identifier is the same as stored in first hardware 910 of FIG. 9.

[0060] Paths 1714 and 1716 provide the electrical connections between:

(i) peripheral port 116 and host computer port 112 or
(ii) peripheral port 120 and computer network port 122

via connectors 1704 and 1708, respectively. Path 1714 and 1716 permit processor 802 to access the contents of first hardware 1710. Inserted, however, in path 1716 of network/host computer-side adapter 1706 is second hardware 1712. Second hardware 1412 disables communication to/from peripheral 114 or 118. It comprises hardware that, by default, enables electrical communication between processor 802 and one of peripherals 114 and 118.

[0061] Second hardware 1712 is under the control of processor 802. Processor 802 retrieves the contents of first hardware 1710 either upon:

(i) boot up of host computer 106;
(ii) attachment of peripheral 118 to network port 122; or,
(iii) attachment of peripheral 114 to host computer port 112.

Processor 802 then compares the unique identifier with a list of identifiers of authorized peripherals. If a match is not found, processor 802 sends commands to second hardware

1712 to disable electrical communication between one of processor 802 and network 102 and one of peripheral 114 and 118. Second hardware 1412 can comprise, for example, one or more normally-closed relays, semiconductor switches, etc., and line-interrupt control. It will be appreciated by those skilled in the art how to make and use second hardware 1712.

[0062] FIG. 18 depicts the contents of memory 804 of the fourth illustrative embodiment of the present invention. In the second illustrative embodiment, memory 804 comprises first software module 1802 comprising a list of identifiers associated with peripherals that are authorized to be connected to host computer 106 or computer network 102; and second software module 1804 for retrieving the contents of first hardware 1410 and taking actions based upon said contents. First software module 1802 has the same contents as first software module 1002 of the first illustrative embodiment.

[0063] FIG. 19 is a flowchart of the actions taken by processor 802 as represented by the contents of second software module 1804. At task 1902, processor 802 retrieves the contents of first hardware 1710 either upon:

(i) boot up of host computer 106;
(ii) the attachment of peripheral 118 to network port 122; or
(iii) the attachment of peripheral 114 to host computer port 112.

[0064] At task 1904, processor 802 compares the unique identifier with a list of identifiers contained within software module 1802.

[0065] At task 1906, processor 802 decides whether a match is found in the list.

[0066] If a match is found, no action is taken (second hardware 1712 by default enables communication with peripherals). If a match is not found, at task 1908, processor 802 sends commands directly to second hardware 1712 to disable communication. Optionally, tasks 1910 and 1912 are executed.

[0067] At task 1910, processor 802 stores information indicative of a failure to find a match of the retrieved unique identifier to elements in the list of software module 1802. This information can take the form of logging the attempt in memory 804, what identifier was retrieved, and supporting data for an investigation.

[0068] At task 1912, processor 802 generates an email indicative of a failure to find a match of the retrieved unique identifier retrieved to elements in the list of software module 1902. The email can be sent to the computer terminal of security officer if computer network 102 is a classified local area network, or to a system administrator.

[0069] It is to be understood that the above-described embodiments are merely illustrative of the present invention and that many variations of the above-described embodiments can be devised by those skilled in the art without departing from the scope of the invention. It is therefore intended that such variations be included within the scope of the following claims and their equivalents.